Reports Methodology

Rob R

Friday, April 17, 2015

## Programming Methodology for KHIIS Reports

### SQL

#### Keys, Constraints, and Utility Functions:

The "Raw Report Tables" file has been left intact for testing purposes. Use the modular form in production, notably field ordering for gallbladder procedure tables differ between the two versions, and down-stream processing relies on the newer ordering within Modular\_Report\_Call.

In all cases, the derived primary key (UniqID) is a concatenation of the following fields: MembershipID, PatientDOB, PatientGenderCode, and FamilyMembershipID.

Selections were constrained by ensuring Member, Summary, and Detail fields: Eligibility End Date, Last Date of Service, and Service Date respectively were within the yearly date range. E.g. 2010-01-01 to 2010-12-31.

All uknown genders (PatientGenderCode=U) were removed.

#### Utility Functions:

* ElgMoMid: Takes the Member record's Eligibility Start Date, End Date, Submission Quarter, and the given year's date. Tests for date reversal (start > end) and fixes if needed. Returns a count of months covered in the quarter. "Covered"" is defined as having coverage as of the 15th of the month. Note: Current method misses some corner cases, e.g. member has no record for quarter 1, but the quarter 2 record back-dates to the start of quarter 1, currently would only count 3 months instead of 6.
* age\_band: Takes Date of Birth (PatientDOB), and Year Start Date. Calculates age as of January 1 of given year and returns age-band assignment of 0-18, 19-64, and 65+.
* elg\_band: Takes a sum of covered months and returns the eligibility band.

|  |  |
| --- | --- |
| Count Months | Band |
| Months < 1 | 0 |
| Months < 4 | 1-3 |
| Months < 7 | 4-6 |
| Months < 10 | 7-9 |
| Months < 12 | 10-12 |
| Months > 12 | 10-12 |

#### Temporary Table Generation ReportProc(Year\_Start='YYYYMMDD',Year\_End='YYYYMMDD'):

Temporary table generation is in a stored procedure (ReportProc) accepting date bounds (year start, and year end).

1. Table dentoptclaims: Created in order to later remove dental and vision claims since not all plans provide such coverage. Derived from Detail file, and filtered on Provider Specialty and Taxonomy codes. See Appendix A (pg 230) and Appendix C (Pgs 249,252-3) of the Data Dictionary. (4th Ed.)

* CREATE TEMPORARY TABLE IF NOT EXISTS   
   dentoptclaims ( INDEX(uniqID,ClaimNumber) )   
  ENGINE=InnoDB   
  AS(  
  SELECT   
   DISTINCT  
  Concat(MembershipID,PatientDOB,PatientGenderCode,FamilyMembershipID) as uniqID,  
  ClaimNumber,  
  Servicedate  
  FROM khiisdetailflatfile  
  WHERE Servicedate BETWEEN @YEAR\_STR AND @YEAR\_END   
  AND   
  (ProviderSpecialtyCode IN('DTA','DTH','DTL','DENT','OPH','OPT','OD','OTT')  
  OR   
  TaxonomyCode IN('126800000X', '124Q00000X', '126900000X', '1223D0001X',   
  '122300000X', '1223E0200X', '1223G0001X', '1223P0106X', '1223X0008X',   
  '1223S0112X', '1223X0400X', '1223P0221X', '1223P0300X', '1223P0700X', '122400000X',  
  '152WC0802X','152WL0500X', '152WX0102X','152W00000X', '152WP0200X',   
  '152WS0006X', '152WV0400X','156FC0800X', '156FC0801X', '156FX1700X',   
  '156FX1100X', '156FX1101X', '156FX1800X', '156FX1201X', '156FX1202X',   
  '156FX1900X','156F00000X')  
  )  
  );

1. Table adjclaims: Created in order to later remove any claims with positive or negative adjustments. Derived from Detail file, selecting all records with ClaimAction adjustments NA and PA.

* CREATE TEMPORARY TABLE IF NOT EXISTS   
   adjclaims ( INDEX(uniqID,ClaimNumber,ServiceDate) )   
  ENGINE=InnoDB   
  AS(  
  SELECT   
   DISTINCT  
  Concat(MembershipID,PatientDOB,PatientGenderCode,FamilyMembershipID) as uniqID,  
  ClaimNumber,  
  Servicedate,  
  ClaimActionType, PlanType,ProductType  
  FROM khiisdetailflatfile  
  WHERE (  
   (Servicedate BETWEEN @YEAR\_STR AND @YEAR\_END)   
   AND   
   (ClaimActionType IN('NA','PA'))  
   )  
  );

1. Table tempyearlytable: An intermediate table to subset Summary Claims to be within year bounds and excludes supplementary and ancillary plans based on Plan Type and Product Type fields. (PlnTyp!=5,6 and PrdTyp=1).

* CREATE TEMPORARY TABLE IF NOT EXISTS   
   tempyearlytable ( INDEX(uniqID,ClaimNumber) )   
  ENGINE=InnoDB   
  AS(  
  SELECT   
   Concat(MembershipID,PatientDOB,PatientGenderCode,FamilyMembershipID) as uniqID,  
   ClaimNumber,  
   ...,  
   ...  
  FROM khiissummaryflatfile  
  WHERE LastDateofService BETWEEN @YEAR\_STR AND @YEAR\_END  
  AND PlanType NOT IN('5','6') AND ProductType='1'  
  );

1. Table yearlytable: Uses intermediate table above and appends binary diagnosis indicators for Diabetes, CHF, COPD, and Asthma. Also LEFT JOINs on dentoptclaims and returns only records not found with dental and vision claims.

* CREATE TEMPORARY TABLE IF NOT EXISTS  
  yearlytable ( INDEX(uniqID,ClaimNumber,FirstDateOfService,LastdateOfService) )   
  ENGINE=InnoDB   
  AS(  
   SELECT a.\*,  
   IF((PrimaryDiagnosis REGEXP '^250.\*')|(SecondaryDiagnosis REGEXP '^250.\*')|(ThirdDiagnosis REGEXP '^250.\*'),1,0) as 'diab',  
   ...,  
   ....  
   FROM tempyearlytable as a  
  LEFT JOIN dentoptclaims as b  
  ON (a.uniqID = b.uniqID  
   AND   
   a.ClaimNumber= b.ClaimNumber  
   )  
  WHERE b.ClaimNumber IS NULL  
  );

1. Table tempmemb: An intermediate table subsetting the Member file to be within year bounds. Also calls age\_band and ElgMoMid functions to calculate member age and coverage based on quarter. Joins Member Zipcode (Truncated at 5 digits) with Zipcode table to return County.

CREATE TEMPORARY TABLE IF NOT EXISTS   
 tempmemb ( INDEX(uniqID) )   
ENGINE=InnoDB   
AS (  
 SELECT   
 NAICNO,  
 Concat(MembershipID,PatientDOB,PatientGenderCode,FamilyMembershipID) as uniqID,  
 ...,  
 age\_band(khiismemberflatfile.PatientDOB, @YEAR\_STR) as AgeBand,  
 ...,  
 LEFT(TRIM(khiismemberflatfile.Zipcode),5) as 'fivezip',  
 b.countynm,  
 b.statename,  
 ...,  
 ElgMoMid(khiismemberflatfile.EligibilityStartDate,  
 khiismemberflatfile.EligibilityEndDate,  
 khiismemberflatfile.SubmissionQtr,@YEAR\_STR) as ElgMoMid,  
 ...,  
 ...  
 FROM  
 khiis.khiismemberflatfile  
 LEFT JOIN zip as b  
 ON(LEFT(TRIM(khiismemberflatfile.Zipcode),  
 5)=b.zip)  
 WHERE  
 EligibilityEndDate BETWEEN @YEAR\_STR AND @YEAR\_END   
);

1. Table memb\_summ: Table summarizing the tempmemb table to get a total sum of months per member, and filter by state, plan and product type.

CREATE TEMPORARY TABLE IF NOT EXISTS   
 memb\_summ ( INDEX(uniqID) )   
ENGINE=InnoDB   
AS (  
 SELECT   
 uniqID,  
 ...,  
   
 SUM(ElgMoMid) as SmElgMoMid,  
 ...,  
 ...  
 FROM  
 tempmemb  
 WHERE statename='Kansas'  
 AND PlanType NOT IN('5','6') AND ProductType='1'  
 AND SubmissionYear=LEFT(@YEAR\_STR,4)  
GROUP BY uniqID);

1. Table claim\_summ: Table summarizes previous yearly table. E.g. calls MAX(diab), and SUM(TotalCharges) with GROUP BY uniqID to get a single member record with diagnosis variables and an aggregate charges over the year. The yearlytable is LEFT JOINed with the adjusted claims table (adjclaims) on uniqID and either Claim Number or if Detail Service Date is within Summary's First and Last Date of Service. Claims matched in the adjclaims table are removed.

CREATE TEMPORARY TABLE IF NOT EXISTS   
 claim\_summ ( INDEX(uniqID,FirstDateOfService,LastdateOfService) )   
ENGINE=InnoDB   
AS (SELECT   
 yearlytable.uniqID,  
 ...,  
 MAX(diab) as diab,  
 MAX(chf) as chf,  
 MAX(copd) as copd,  
 MAX(asthm) as asthm,  
 CONVERT(SUM(yearlytable.TotalCharges)/100,Decimal(11,2)) as SmTotChg,  
 CONVERT(SUM(yearlytable.TotalAllowed)/100,Decimal(11,2)) as SmTotAllwd,  
 CONVERT(SUM(yearlytable.TotalPaid)/100,Decimal(11,2)) as SmTotPd,  
 CONVERT(SUM(yearlytable.MemberResponsibility)/100,Decimal(11,2)) as MbrRsp,  
 ...,  
 ...  
 FROM  
 khiis.yearlytable  
 LEFT JOIN  
 adjclaims as b ON (  
 yearlytable.uniqID = b.uniqId  
 AND yearlytable.PlanType = b.PlanType  
 AND (  
 (yearlytable.Claimnumber=b.ClaimNumber) OR   
 (b.servicedate BETWEEN yearlytable.FirstDateOfService AND yearlytable.LastDateOfService)  
 )  
 )  
 WHERE  
 b.ClaimNumber IS NULL  
GROUP BY uniqID);

1. Temporary tables for specific procedures -- Hip Replacement, Colonoscopy etc. are then created based on CPT procedure codes in the Detail file. Only Major Med Plans are included. (PlnTyp!=5,6 and PrdTyp=1).

CREATE TEMPORARY TABLE IF NOT EXISTS   
 gallbladder ( INDEX(uniqID,ClaimNumber) )   
ENGINE=InnoDB   
AS(  
SELECT   
 DISTINCT  
 Concat(MembershipID,PatientDOB,PatientGenderCode,FamilyMembershipID) as uniqID,  
 ClaimNumber,  
 Servicedate,  
 RevenueProcedureCode  
FROM khiisdetailflatfile  
WHERE Servicedate BETWEEN @YEAR\_STR AND @YEAR\_END   
AND PlanType NOT IN('5','6') AND ProductType='1'  
AND RevenueProcedureCode IN('47562','47563','47564')  
);

#### Aggregated Outputs - Modular\_Report\_Call:

1. *Year* Raw Members: Gives uniqID, Age Band, Patient Gender, County, Months Eligibility Band, Months Eligible, and submission year.

SELECT   
 uniqID,  
 AgeBand,  
 PatientGenderCode as Gender,  
 countynm as County,  
 elg\_band(SmElgMoMid) as ElgBand,  
 SmElgMoMid,  
 SubmissionYear  
FROM  
 memb\_summ  
WHERE PatientGenderCode IN ('M','F')  
ORDER BY countynm  
  
INTO OUTFILE 'D:\\MySQL\_KHIIS\_OUT\\2012\_Raw\_Members.csv'  
FIELDS TERMINATED BY ','  
LINES TERMINATED BY '\n'  
;

1. *Year* Raw Claims: Combines summarized claims information (e.g yearly total of AllowedCharges) and county and eligibility band information from the Member records.

SELECT   
 a.uniqID,  
 AgeBand,  
 a.PatientGenderCode,  
 countynm AS County,  
 elg\_band(SmElgMoMid) as ElgBand,  
 SmElgMoMid as MembMo,  
 SmTotChg,  
 SmTotAllwd,  
 SmTotPd,  
 MbrRsp,  
 diab,  
 chf,  
 copd,  
 asthm,  
 a.SubmissionYear  
   
FROM  
 memb\_summ as a JOIN claim\_summ as b ON  
 (a.uniqID=b.uniqID)  
WHERE a.PatientGenderCode IN ('M','F')  
  
INTO OUTFILE 'D:\\MySQL\_KHIIS\_OUT\\2012\_Raw\_Claim\_Summ.csv'  
FIELDS TERMINATED BY ','  
LINES TERMINATED BY '\n'

1. *Year* *Procedure* Claims: Procedure level claim summaries are created by combining the previously created procedure tables with subset Summary table yearlytable. Summary claims are selected by claim or date range and then grouped by uniqID and summed to get a total incident cost-of-care measure.

SELECT   
 b.uniqID,  
 ...,  
 MAX(diab) as diab,  
 MAX(chf) as chf,  
 MAX(copd) as copd,  
 MAX(asthm) as asthm,  
 CONVERT(SUM(b.TotalCharges)/100,Decimal(11,2)) as SmTotChg,  
 CONVERT(SUM(b.TotalAllowed)/100,Decimal(11,2)) as SmTotAllwd,  
 CONVERT(SUM(b.TotalPaid)/100,Decimal(11,2)) as SmTotPd,  
 CONVERT(SUM(b.MemberResponsibility)/100,Decimal(11,2)) as MbrRsp,  
 ...,  
 ...  
 FROM gallbladder as a LEFT JOIN yearlytable as b ON a.uniqID=b.uniqID  
AND   
(  
 (a.servicedate between b.FirstDateofService and b.LastDateofService)  
)  
JOIN memb\_summ as c ON a.uniqID=c.uniqID  
WHERE b.PatientGenderCode IN ('M','F')  
GROUP BY b.uniqID  
  
INTO OUTFILE 'D:\\MySQL\_KHIIS\_OUT\\2012\_Gallbladder\_Claim\_Summ.csv'  
FIELDS TERMINATED BY ','  
LINES TERMINATED BY '\n'  
;

### R

#### Libraries:

data.table, lubridate, dplyr, stringr, assertthat, reshape2.

#### Definition:

In our calculations Average Total Cost is defined as the sum of total paid charges and member responsiblity for all members of the group divided by the sum of group member months divided by twelve or for each group SUM(TotPd+Mbrsp)/(SUM(MembMo)/12). If the sum of member months was 0, the square root of the group size was used instead or SUM(TotPd+Mbrsp)/sqrt(Group Size).

#### Style:

Coding primarily makes use of the split-apply-combine paradigm and R's "pipe" operator %>% which should be considered similar to a unix '|' or an arrow showing directed flow.

#### Utility and Helper Functions:

All reading functions have a hard-coded base path and file name suffix. E.g.

base\_path = 'D:/MySQL\_KHIIS\_OUT/'  
file\_name\_suffix = '\_Raw\_Members.csv'

Source: readData.r Functions:

* readMembersData(Year) - Read in members data, set header/column names appropriately.
* readClaimsData(Year) - Read in claims data, set header/column names, create "TotCost" variable which is the sum of Total Allowed Charges and Member Responsibility.
* readProcedureData(Year,Procedure) - Read in Procedure data, set names, create "TotCost" variable. Procedure must be one of: Hip,Gall,Mammo or Colo.

Soure: PopEst.R  
Functions: Reads in and calculates adjusted population for each county based on yearly estimated population - general quarters population. 2013 vintage intercensal estimates used, see file for more details.

#### Final Report Generation:

1. Membership Analysis  
   Source: Membership\_Analysis.R  
   Reads member data and creates counts and summaries based on grouping by age band, eligibility band, county, gender, and year. Fills any missing variable combinations with 0's and writes to disk. Also creates Completeness estimates by calculating percent of estimated county population that is found within KHIIS member file. Assigns to quartiles C1 - C4 and writes to disk.
2. Total Cost of Care  
   Source: Tot\_Cost\_Care.R  
   Creates full data frame of claims data for all years. Groups data by gender, county, age band, eligibiity band, and year. For each group, calculates median TotCost and average TotCost. Any missing group combination is created and filled with 0s. The table is written to disk. Subsequent variations omit certain groups in order to derive other tables -- e.g. median cost of care in a county and year without splitting by gender. E.g. assume all calims have been read in and put in a table or dataframe 'tmp' for all years. The next line takes the table directs it to the group\_by function, which designates the following variables to group on.

tmp %>% group\_by(gender,county,ageband,elgband,subyr) %>%

The result is passed or piped to the summarise verb which splits by the cartesian product of all selected grouping vriables E.g. M, Johnson County, 0-19, 1-3, 2010 etc. On each grouping it creates new columns based on the calculation functions passed.

summarise(Med = median(TotCost),  
 AvgTot=  
 ifelse( sum(membmo)==0,  
 sum(TotCost)/sqrt(n()),  
 sum(TotCost)/(sum(membmo)/12)  
 ),  
 ClassTotal=sum(TotCost),ClassMemMo=sum(membmo),ClassSize=n())

The above lines calculate the median per grouping of total cost "Med", as well as an Average total cost "AvgTot". The ifelse performs the Average Total calculation as previously described and is a shortened form of IF(TRUE) THEN do A, ELSE do B. The "n()" returns the number of rows/entries in the given grouping. (A number of variable type-casts have been removed for readability.)

1. Diagnosis Cost of Care  
   Source: Diagnosis\_Cost\_Care.R, Diagnosis\_gndrVars\_Cost\_Care.R  
   Follows the same pattern as Total Cost of Care with groupings on Disease binary variables. Calculates median and average cost of care with disease groupings. The gendered version is not currently used as it identifies individuals too specifically.
2. Procedure Cost of care  
   Source: Procedure\_Cost.R  
   Follows Total Cost of Care pattern reading in procedure based tables and returning median and mean cost of care for various groupings.